rxInventory Benutzerhandbuch

Rubinox GmbH

<contact@rubinox.com>

rxInventory Benutzerhandbuch

Rubinox GmbH
<contact@rubinox.com>

Version 3.7.25.0

Veröffentlicht 2015-07-07

Zusammenfassung

Benutzerhandbuch für die rxInventory Netzwerkinventarisierung.

1. Installation 1
1. Systemvoraussetzungen 1
2. Ausführen der Installation 1
3. Konfiguration der Datenbank 1
3.1. Microsoft Jet Engine 2
3.2. Microsoft SQL Server 2
3.3. Microsoft SQL Server Compact Edition 3.5 2
3.4. Oracle 2
3.5. MySQL 3
3.6. PostgreSQL 3
2. Ermittlung von Geräten im Netzwerk 4
1. Scangruppen 4
1.1. Sammelverzeichnisse 4
2. Konfiguration einer Scangruppe 5
2.1. Allgemein-Reiter 5
2.2. Skripte-Reiter 6
2.3. Erweitert-Reiter 6
3. Scan-Planer 7
3. Umgang mit den Inventarisierungsdaten 8
1. Geräteklassen 8
2. Eigene Assets 9
3. Liste der Softwarepakete 9
4. Archiv 9
5. Benutzerdefinierte Eigenschaften 9
6. Das Journal 10
7. Berichtsdruck und Datenexport 11
8. Eigene Berichte 11
8.1. Filter und Having 12
8.2. Verlauf 12
8.3. SQL Views 12
9. Berichtsbäume 13
10. Web Interface 13
4. Software-Lizenzmangement 15
1. License Packs 16
2. Softwareregeln 16
3. Lizenzen Computern zuweisen 16
5. Eigene Skripts 18
1. Die Programmiersprache 18
2. Struktur der Skripts 18
3. VB-Funktionsreferenz 20
3.1. Ausgabefunktionen 20
3.2. Windows-Informationsfunktionen 21
3.3. WMI-Funktionen 22
3.4. Registryfunktionen 22
3.5. Dateifunktionen 24
3.6. Active Directory-Funktionen 26
3.7. SNMP-Funktionen 27
3.8. SSH-Funktionen 27
3.9. VB-Funktionen 28
4. Spezielle Funktionen 32
4.1. NewDictionary 32
5. Callback-Funktionen 32
5.1. Eintrags-Callbacks 32
6. Automatisierung von Vorgängen 35
1. InvCmd.exe Kommandozeilenparameter 35

Kapitel 1. Installation

1. Systemvoraussetzungen

- Windows 2000 oder neuer
- 512MB Arbeitsspeicher
- .NET Runtime 2.0 oder höher

rxInventory nutzt standardmäßig die Microsoft Jet Engine (JET) um die Inventardaten zu speichern. JET ist auf allen unterstützten Betriebssystem vorinstalliert und wird auch von Microsoft Access genutzt. Falls Sie auf die Daten von mehreren Rechnern aus zugreifen oder Sie mehr als ein paar hundert Geräte in der Datenbank verwalten möchten, sollten sie rxInventory mit einem SQL-Server verbinden.

Die folgenden SQL-Datenbanken werden von rxInventory unterstützt.

- Microsoft Sql Server 2000 und höher
- Microsoft Sql Server Compact Edition 3.5
- Oracle 9i, Oracle Database 10g, Oracle Database 11g, Oracle Database 12c
- MySQL Server 5.x, MariaDB 5.x und 10
- PostgreSQL Server 8.x and 9.x

Für MySQL/MariaDB und Oracle-Datenbanken muss zuerst ein passender ADO.NET-Treiber installiert werden.

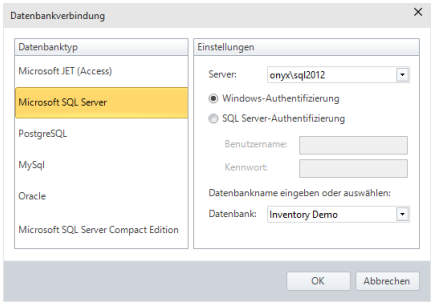
2. Ausführen der Installation

Die neueste Version von rxInventory kann von http://www.rxinventory.net [http://www.rxinventory.net>] heruntergeladen werden. Für eine erfolgreiche Installation sind Administratorrechte erforderlich. Ein einfacher Doppelklick auf die rxInventory.msi-Datei startet den Installationsvorgang. Ist eine alte Version von rxInventory bereits vorhanden, wird diese automatisch auf den aktuellen Stand gebracht.

Nach der Installation können Sie die rxInventory-Konsole vom Startmenü aus aufrufen.

3. Konfiguration der Datenbank

Die Standarddatenbank liegt im `All Users`-Profil. Der Speicherort der `.mdb`-Datei kann durch einen Klick des Datenbank-Links auf der Hauptseite eingestellt werden. Sie können sich dort auch zu einer SQL-Datenbank Ihrer Wahl verbinden. Falls die Datenbank noch keine rxInventory-Tabellen enthält, werden diese automatisch erzeugt.



Anmerkung

Beim Wechseln der Datenbank werden schon vorhandene Daten nicht automatisch kopiert. Sie müssen deshalb die Daten der aktuellen Datenbank erst sichern und dann in der neuen wiederherstellen. Falls Sie nur die JET-Datenbank an einen anderen Ort verschieben möchten, können Sie die Datei alternativ auch einfach kopieren.

3.1. Microsoft Jet Engine

Die Microsoft Jet Engine ist Bestandteil des Betriebssystems. Sie ist ausreichend schnell und funktioniert gut bis zu einigen hundert Geräten in der Datenbank. Wenn eine nicht vorhandene Datei angegeben wird, wird eine leere Datenbank angelegt.

Warnung

Auch wenn es in einigen Fällen funktionieren mag, so empfehlen wir nicht die `.mdb`-Datei über eine Netzwerkfreigabe anzubinden. Die Datenbank kann beschädigt werden, wenn sie über das Netzwerk bearbeitet wird und dieses dabei unterbrochen wird.

3.2. Microsoft SQL Server

Der Microsoft SQL Server ADO.Net Provider ist in der .NET-Runtime integriert. Es ist keine zusätzliche Installation nötig.

3.3. Microsoft SQL Server Compact Edition 3.5

Warnung

Mit Version 3.7 von rxInventory unterstützen wir die Microsoft SQL Server Compact Edition nicht mehr. Sie ist nur aus Gründen der Rückwärtskompatibilität noch vorhanden. Bitte stellen Sie auf eine andere Datenbank um.

3.4. Oracle

Um auf eine Oracle-Datenbank zuzugreifen, muss der Oracle Data Provider für .NET (ODP.NET) installiert werden. Er kann von http://www.oracle.com heruntergeladen werden. Falls der Provider fehlt oder nicht richtig installiert ist, bekommen Sie eine Fehlermeldung, dass die `Oracle.Data.Access.dll` fehlt.

Zum Verbinden einer Oracle-Datenbank müssen Sie den Server-Namen **und** den Namen der Datenbankinstanz getrennt durch einen umgekehrten Schrägstrich (Backslash) angeben. Zum Beispiel `myoraserver/orcl`. Ein Eintrag in die `TNSNAMES.ORA`-Datei ist nicht nötig.

3.5. MySQL

Um auf eine MySQL-Datenbank zuzugreifen muss der MySQL `Connector / Net 5.0` installiert sein. Er kann von http://www.myql.com [http://www.mysql.com] heruntergeladen werden. Falls der Provider fehlt oder nicht richtig installiert ist, bekommen Sie eine Fehlermeldung, dass die `MySql.Data.dll` fehlt. Falls Sie die binaries-only `zip`-Datei von MySQL herunterladen, genügt es eine Kopie der MySql.Data.dll in das rxInventory-Installationsverzeichnis zu kopieren.

3.6. PostgreSQL

Der PostgreSQL ADO.Net-Provider ist in der rxInventory-Installation integriert. Es wird keine zusätzliche Installation benötigt.

Kapitel 2. Ermittlung von Geräten im Netzwerk

1. Scangruppen

Eine Scangruppe legt fest, wie und welche Geräte gescannt werden sollen. Jede Scangruppe besteht aus einer oder mehreren Scanbereichen. Ein Scanbereich kann einer der folgenden Typen sein:

Host oder IP-Adresse	Legt einen einzelnen Rechner durch seinen DNS/NetBIOS-Namen oder seine IP-Adresse fest.
IP-Bereich	Ein IP-Bereich wird durch die erste und letzte Adresse definiert. Alle gültigen IP-Adressen in dem Bereich werden durchprobiert, inklusive der ersten und der letzten Adresse. Es müssen keine Vorkehrungen getroffen werden, um Broadcast- (z.B. 192.168.1.255) oder Netzwerk-Adressen (z.B. 192.168.1.0) auszuschließen.
Sammelverzeichnis	Legt einen Dateisystemordner fest, aus dem Scandateien eingesammelt werden.

Beim ersten Start von rxInventory, wird eine Scangruppe mit dem Namen `Standard` angelegt und ein einzelner IP-Bereich wird hinzugefügt, der das lokale Subnetz umfasst. Sie können weitere Scanbereiche hinzufügen, bearbeiten oder löschen.

1.1. Sammelverzeichnisse

Neben dem entfernten Scannen über das Netzwerk kann ein Windows Rechner auch dadurch inventarisiert werden, dass ein kleines Programm (`rxscan.exe`) auf ihm ausgeführt wird. Es kann entweder über ein Benutzer-Logonskript gestartet werden oder durch eine Active Directory-Gruppenrichtlinie. Der Vorteil dieser Methode ist, dass die Abhängigkeit von der Netzwerkkonfiguration (Firewall, Fernzugriff auf WMI usw.) geringer ist als bei einem Remote-Scan. Es funktioniert in fast jeder Umgebung, selbst wenn keine Netzwerkverbindung vorhanden ist. (z.B. durch Einsatz eines USB-Sticks um Inventarisierungsdaten zu sammeln)

Beim Anlegen eines `Sammelverzeichnis`-Scanbereichs muss ein Dateisystemordner angegeben werden. In den meisten Fällen sollte es sich dabei um eine Netzwerkfreigabe handeln, die für alle Benutzer zugreifbar und schreibbar ist. Das Scanprogramm (`rxscan.exe`) und seine Konfigurationsdatei (`rxscan.ini`) werden automatisch in dem Verzeichnis angelegt. Die Konfigurationsdatei enthält die Einstellungen der Scangruppe und wird jedes Mal upgedatet, wenn die Konfiguration der Scangruppe geändert wird. `rxscan.exe` sucht selbstständig nach der Konfigurationsdatei in seinem Verzeichnis und erzeugt eine Inventarisierungsdatei. Der Dateiname der erzeugten Datei hat die Erweiterung `.rxs` und besteht aus dem Computernamen und dem aktuellen Zeitstempel.

Anmerkung

Sie können die beiden Dateien an einen anderen Ort kopieren (z.B. USB-Stick) und sie von dort ausführen. Damit rxInventory die neu erzeugten Inventarisierungsdateien einlesen kann, müssen Sie in das ursprüngliche Sammelverzeichnis zurückkopiert werden. Bitte achten Sie drauf, dass Sie die Konfigurationsdatei und das Scanprogramm erneut kopieren, immer wenn sich die Konfiguration ändert oder rxInventory upgedatet wurde.

2. Konfiguration einer Scangruppe

Jede Scangruppe hat ihre eigenen Konfigurationseinstellungen, die die Inventarisierung der Geräte beeinflussen.

2.1. Allgemein-Reiter

Windows Benutzerkennung

Beim Erfassen eines Windows-Rechners über das Netzwerk wird ein Benutzerkonto benötigt, das dort Administratorrechte hat. Standardmäßig benutzt rxInventory die integrierte Windows-Authentifizierung, was bedeutet, dass der aktuell angemeldete Benutzer für die Authentifizierung verwendet wird. Diese Art der Authentifizierung sollte in den meisten Fällen funktionieren, bei denen in einer einzelnen Domain oder einem Active Directory inventarisiert wird.

In komplizierteren Umgebungen, wie Windows-Domänen ohne Vertrauensstellung oder Windows-Arbeitsgruppen, kann es nötig sein explizit einen Benutzernamen und Passwort anzugeben um auf die Windowsgeräte im Netzwerk zuzugreifen.

Hier sind einige Beispiele für Benutzernamen:

`mydomain\myadministrator`	Dies ist die gewöhnliche Syntax für Windows-Domänen.
`admin@mydomain.com`	Der User-Principal-Name (UPN)
`myadministrator`	Sie können den Domänennamen weglassen, wenn sie einen lokalen Administrator auf dem zu scannenden Computer angeben wollen. Zum Beispiel, wenn das Gerät nicht Teil einer Domäne ist.

Wie andere Konfigurationseinstellungen einer Scangruppe, wird der Benutzername in der Datenbank gespeichert. Das Passwort dort auch zu speichern, würde ein Sicherheitsrisiko darstellen, falls auch andere Leute auf die Datenbank zugreifen können. Aus diesem Grund wir das Passwort immer lokal auf dem Rechner gespeichert, auf dem es eingegeben wurde. Deshalb müssen Sie, falls Sie mehrere Computer haben von denen aus Sie scannen möchten, das Passwort auf jedem dieser Computer erneut eingeben.

Die Benutzerkennung kann auf der Ebene eines Scanbereichs bzw. Hosts explizit anders gesetzt werden.

SSH-Benutzer

Linux- und Mac-Computer können über das SSH-Protokoll erfasst werden. Dafür muss hier ein Benutzername und Passwort angegeben werden. Fall kein Name angegeben wurde, wird das SSH-Protokoll überhaupt nicht verwendet.

Der SSH-Benutzer kann auf der Ebene eines Scanbereichs bzw. Hosts explizit anders gesetzt werden.

Anmerkung

Auf Linux-Betriebssystem können einige Hardwareinformationen nur abgefragt werden, wenn man mit Root-Privilegien angemeldet ist. Das können Sie entweder mit dem Root-Benutzer erreichen, was nicht emfehlenswert ist, oder Sie erlauben einem normalen Benutzer `sudo` aufzurufen indem Sie die `/etc/sudoers`-Datei editieren. Stellen Sie bitte sicher, dass die `requiretty`-Option für diesen Benutzer **nicht** gesetzt ist.

SNMP-Communities

rxInventory verwendet das SNMP-Protokoll um Netzwerkgeräte zu ermitteln. Wenn keine SNMP-Community angegeben wurde, wird `public` als Standard genommen. Sie können mehrere SNMP-Communities angeben, indem sie diese mit Kommas trennen. Während dem Scan werden sie in dieser Reihenfolge durchprobiert.

2.2. Skripte-Reiter

Scanskripte ermöglichen die Menge der Eigenschaften zu erweitern, die von einem Gerät erfasst werden. Beim Drücken des `Neu`-Buttons können Sie unter einer Anzahl von Vorlagen wählen. Der `Test`-Button überprüft die Syntax des Skripts und führt es gegen den lokalen Computer aus. Die daraus resultierenden Informationen werden in einer Dialogbox angezeigt, aber nicht zu den Inventarisierungsdaten hinzugefügt bis ein normaler Scan gelaufen ist.

Dokumentation zum Skriptsyntax und den unterstützten Funktionen finden Sie unter Kapitel 5, *Eigene Skripts*.

2.3. Erweitert-Reiter

Maximale Zahl gleichzeitiger Scans	Beim Scannen über eine langsame Netzwerkverbindung können Sie die Zahl der Scans reduzieren. Falls Sie einen großen IP-Bereich haben, der nur schwach mit Geräten besetzt ist, können Sie das Limit heraufsetzen.
Externe Scanprozesse verwenden	Standardmäßig verwendet rxInventory interne Threads um das Netzwerk abzufragen. Wenn diese Option eingeschaltet ist, wird diese Aufgabe durch ein externes Programm `rxscan.exe` ausgeführt. Das verlangsamt den Scanprozess, aber kann die Suche nach Problemen bei einer Fehlfunktion erleichtern.
Scandateien speichern	Dies ist ebenfalls eine Option zur Fehlersuche. Für jedes Gerät das gescannt wird, wird eine Datei in dem lokalen Anwendungsdatenverzeichnis angelegt. z.B. `c:\Users\<myuser>\AppData\Local\Rubinox\ScanFiles`.
WMI nicht verwenden	Standardmäßig verwendet rxInventory die Windows Management Instrumentation (WMI) um Informationen von einem Computer zu ermitteln. Falls WMI aus irgendeinem Grund nicht zur Verfügung steht, wird ersatzweise auf die Registry zugegriffen. In diesem Fall werden weniger Hardwaredaten erfasst. Durch diese Option können Sie die Verwendung der Registry anstatt WMI erzwingen. Das erhöht die Scangeschwindigkeit signifikant. Sie kann auch Sinn machen, falls WMI-Verbindungen durch eine Firewall blockiert werden, weil die damit verbundenen Timeouts vermieden werden.
Angemeldete Benutzer nicht ermitteln (Datenschutz)	Mit dieser Option verhindern Sie, dass rxInventory Informationen über den gerade angemeldeten Benutzer speichert. Wenn diese Option eingeschaltet ist und ein Computer erfasst wird, der schon in der Datenbank existiert, werden zusätzlich alle Benutzerdaten aus dem Verlauf des Geräts gelöscht.

SNMP nicht verwenden	Falls Sie nicht an SNMP-Netzwerkgeräten interessiert sind, können Sie diese Option einschalten. Dadurch können Sie einige rxInventory Client-Lizenzen einsparen.
Maximale Zahl von MACs pro Switch-Port	Netzwerk-Switches halten eine Liste aller MAC-Adressen, die an einem Port empfangen wurden. rxInventory liest diese Liste aus und verwendet sie, um die Geräte anzuzeigen, die an einem bestimmten Port angeschlossen sind, und umgekehrt. Ports, die als Uplink zu anderen Switches dienen, lernen zwangsläufig eine große Anzahl an MAC-Adressen, die aber für diese Funktion irreführend sind. Aus diesem Grund hält rxInventory einen Port für einen Uplink-Port, wenn dieser mehr als eine bestimmte Anzahl von MACs erlernt hat. Die Standardeinstellung ist 10 und kann hier angepasst werden.
Additional parameters	Hier können spezielle Optionen für den Scan eingestellt werden. Bitte geben Sie dort nur einen Text ein, wenn Sie von unserem Support-Team dazu aufgefordert wurden.

3. Scan-Planer

rxInventory stellt eine Zeitsteuerung zur Verfügung, die es erlaubt Netzwerkscans zu planen. Der Zeitplandialog wird geöffnet indem man den `Scans planen`-Link auf auf der Hauptseite der Konsole anklickt.

Geplante Scans werden durch den `rxInventoryService`-Dienst ausgeführt. Nach der ersten Installation, läuft der Dienst unter dem lokalen System-Account. Wenn Sie den ersten Scan planen, werden Sie nach einem Benutzernamen und Passwort gefragt, das dem Dienst zugewiesen wird. Fall Sie hier keine Kennungsinformationen eingeben wollen, können Sie das auch manuell innerhalb der Diensteverwaltung von Windows machen.

Anmerkung

Die Dienstkennung wird auch für die integrierte Authentifizierung an den Microsoft SQL Server und für den Zugriff auf Sammelverzeichnisse und andere Ressourcen verwendet.

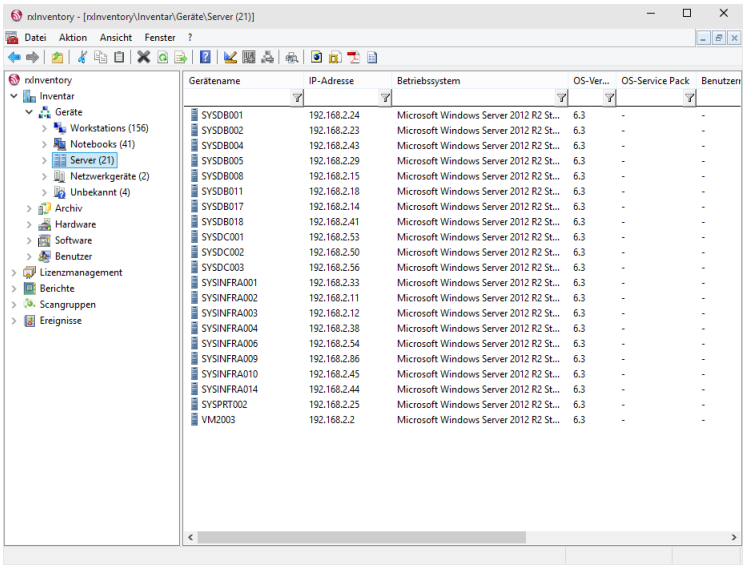
Kapitel 3. Umgang mit den Inventarisierungsdaten

1. Geräteklassen

Die erfassten Geräte können innerhalb des `Inventar`-Unterknotens der rxInventory-Konsole untersucht werden. Ein Gerät, das zum ersten Mal erfasst wird, wird automatisch einer Geräteklasse zugeordnet. (Workstation, Server, Notebook etc.) Sie können die Klasses eines Geräts im Kontextmenü (rechte Maustaste) ändern oder es auf den entsprechenden Klassenknoten ziehen.

Anmerkung

Wenn Sie die Geräteklasse (oder jede andere Eigenschaft, die die Position von Knoten im Baum verschiebt) ändern, dann werden nur der aktuelle Knoten und seine Unterknoten aktualisiert. Deshalb kann es nötig sein, weitere betroffene Knoten manuell zu aktualisieren, um die geänderten Informationen zu sehen. Während ein Ordner ausgewählt ist, kann dies einfach durch Drücken der **F5**-Taste oder durch Auswahl von `Aktualisieren` erreicht werden. Dabei werden auch alle Unterordner aktualisiert.


rxInventory - [rxInventory\Inventar\Geräte\Server (21)]
Datei Aktion Ansicht Fenster ?
rxInventory
Inventar
Geräte
Workstations (156)
Notebooks (41)
Server (21)
Netzwerkgeräte (2)
Unbekannt (4)
Archiv
Hardware
Software
Benutzer
Lizenzmanagement
Berichte
Scangruppen
Ereignisse
Gerätename
IP-Adresse
Betriebssystem
OS-Ver...
OS-Service Pack
Benutzer
SYSDB001 192.168.2.24 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB002 192.168.2.23 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB004 192.168.2.43 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB005 192.168.2.29 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB008 192.168.2.15 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB011 192.168.2.18 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB017 192.168.2.14 Microsoft Windows Server 2012 R2 St... 6.3
SYSDB018 192.168.2.41 Microsoft Windows Server 2012 R2 St... 6.3
SYSDC001 192.168.2.53 Microsoft Windows Server 2012 R2 St... 6.3
SYSDC002 192.168.2.50 Microsoft Windows Server 2012 R2 St... 6.3
SYSDC003 192.168.2.56 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA001 192.168.2.33 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA002 192.168.2.11 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA003 192.168.2.12 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA004 192.168.2.38 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA006 192.168.2.54 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA009 192.168.2.86 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA010 192.168.2.45 Microsoft Windows Server 2012 R2 St... 6.3
SYSINFRA014 192.168.2.44 Microsoft Windows Server 2012 R2 St... 6.3
SYSPRT002 192.168.2.25 Microsoft Windows Server 2012 R2 St... 6.3
VM2003 192.168.2.2 Microsoft Windows Server 2012 R2 St... 6.3

Die folgenden Geräteklassen sind vordefiniert:

Workstations	Computer mit einer Betriebssystemversion für Workstations, an denen kein Akku angeschlossen ist.
Notebooks	Computer mit einer Betriebssystemversion für Workstations, an denen mindestens ein Akku angeschlossen ist.
Server	Computer mit einem Serverbetriebssystem, unabhängig davon ob ein Akku vorhanden ist.
Netzwerkgeräte	Geräte, die das SNMP-Protokoll unterstützen.
Unbekannt	Geräte, die nicht richtig auf die von rxInventory verwendeten Protokolle reagiert haben oder Geräte, für die die Zugriffsrechte durch das Benutzerkonto nicht ausreichend waren.

Eigene Geräteklassen werden über das Kontextmenü auf dem `Geräte`-Knoten angelegt. Sie können dabei zwei Namen und zwei Symbole für die neue Klasse angeben. Das `Singular`-Symbol wird für die Geräteknoten benutzt, die dieser Klasse angehören, und `Plural`-Symbol und -Name werden für den Klassenknoten verwendet, unter welchem die entsprechenden Geräte angezeigt werden. Für Singular und Plural kann das gleiche Symbol angegeben werden.

Im Bearbeitungsdialog der Geräteklassen können auch die Spalten ausgewählt werden, die angezeigt werden, wenn der Geräteklassenknoten selektiert wird.

Im normalen Listenmodus einer Geräteklasse erhalten Sie detaillierte Information über die Hardware und Software, indem Sie auf ein bestimmtes Gerät doppelklicken.

2. Eigene Assets

In rxInventory können eigene Assets von Hand hinzugefügt werden. Das ist zum Beispiel praktisch, wenn Geräte nicht automatisch gescannt werden können, aber in der Datenbank gehalten werden sollen. Eigenen Assets können innerhalb des Context-Menüs einer Geräteklasse angelegt werden. Obwohl eigene Assets innerhalb der vordefinierten Geräteklassen angelegt werden können, ist es sinnvoll eine neue Geräteklasse für sie anzulegen.

Der Name des Assets und einige der meistverwendeten Eigenschaften einschließlich benutzerdefinierter Eigenschaften können bei eigenen Assets gesetzt werden.

3. Liste der Softwarepakete

Unter dem `Inventory->Software`-Knoten gibt es nach Betriebssystemen gruppierte Listen aller Softwarepakete. Unter jedem Softwarepaket werden wiederum alle Computer angezeigt, auf denen das jeweilige Paket installiert ist.

Es ist möglich uninteressante Softwarepakete in der Liste zu verstecken. Um diese versteckte Software zu sehen, kann die Ansicht im Kontextmenü des übergeordneten Knotens umgeschaltet werden.

Die Computerliste unter einem Softwarepaket, kann auch negiert werden. In dem Fall werden nur die Computer angezeigt, auf denen ein bestimmtes Softwarepaket nicht installiert ist.

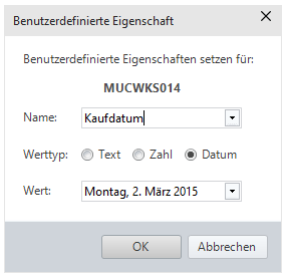
4. Archiv

Computer und Geräte, die nicht mehr in Benutzung sind, die Sie aber nicht aus der Datenbank löschen wollen, können in den Archivknoten verschoben werden. Archivierte Geräte erscheinen **nicht** unter `Software` und `Hardware`, werden nicht von Abfragen zurückgegeben und zählen zu keinen Statistiken. Jedoch benötigen sie eine rxInventory-Lizenz.

5. Benutzerdefinierte Eigenschaften

Neben den Standardeigenschaften können Sie einem Geräte auch manuell eigene Informationen zuweisen. Das geschieht durch das Anlegen sogenannter benutzerdefinierter Eigenschaften. Wenn eine benutzerdefinierte Eigenschaft mit einem bestimmten Namen hinzugefügt wird, wird eine Datenbankspalte

mit dem ausgewählten Typ erzeugt. Danach kann der Datentyp für diese Eigenschaft nicht mehr geändert werden.



Es gibt drei verschiedene Datentypen:

`Text` Der flexibelste Datentyp ist der `Text`-Typ. Er erlaubt einen beliebigen Text mit bis zu 256 Zeichen hinzuzufügen. Der Nachteil ist, dass Zahlen und Datumsangaben nicht richtig sortiert werden können, außer Sie verwenden ein fixes Format.

`Zahl` Der `Zahl`-Typ unterstützt Fließkommazahlen.

`Datum` Aus der Datumsauswahlbox kann ein Datum gewählt werden.

Sie können eine Eigenschaft im Context-Menü des Properties löschen. Der Wert der Eigenschaft wird für dieses Gerät geleert.

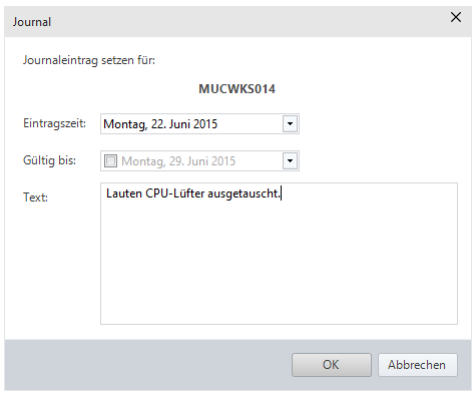
Falls Sie eine Eigenschaft nicht mehr benötigen, können Sie die benutzerdefinierte Eigenschaft auch vollständig für alle Geräte aus der Datenbank entfernen.

Wichtig

Wenn Sie eine Eigenschaft entfernen, wird die zugeordnete Datenbankspalte aus der Datenbank gelöscht. Dieser Vorgang kann **nicht rückgängig** gemacht werden. **Also prüfen Sie sorgfältig, bevor Sie ein Property entfernen!**

6. Das Journal

Jedes Gerät hat ein Journal, in dem Sie Notizen über die Vergangenheit eines Gerätes aufnehmen können. Jede Information, wie Hardwareänderungen oder wann der Leasingvertrag des Computers ausläuft, kann hier gespeichert werden.



Es gibt zwei Datumsangaben. Die Eintragszeit wird standardmäßig auf den aktuellen Tag gesetzt und kann auch geändert werden. Zusätzlich gibt es ein zweites Datum, welches angibt, wie lange der Eintrag gültig

ist. Wenn die Gültigkeit eines Eintrags überschritten ist, wird er mit einem kleinen Ausrufezeichen in der Liste angezeigt. Es gibt eine Beispielabfrage, die alle abgelaufenen Journaleinträge anzeigt.

7. Berichtsdruck und Datenexport

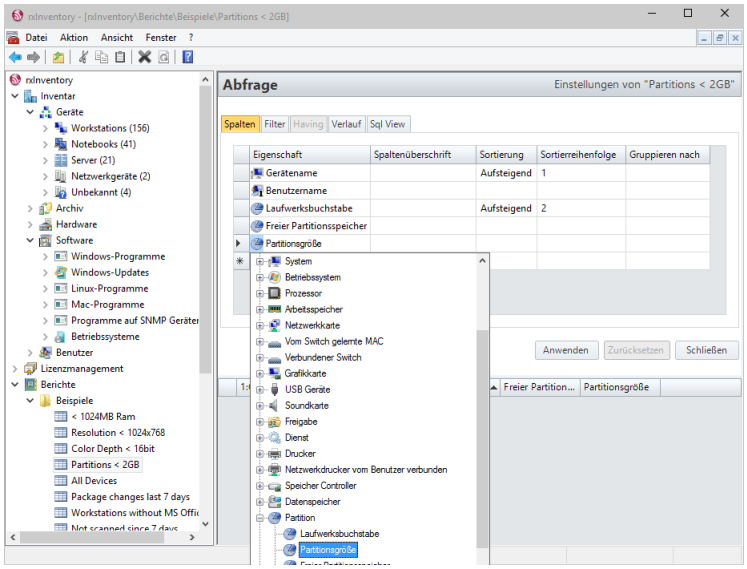
Berichte können von fast jedem rxInventory-Knoten vorangesehen, gedruckt und export werden. Unterstützt werden PDF, HTML, MHT (HTML mit allen notwendigen Dateien zusammengefasst in einer Datei) und CSV.

Falls ein Knoten Unterknoten enthält, werden diese mit eingeschlossen. (Trifft nicht auf CSV-Exporte zu) Um die Anzahl der Seiten auf eine vernünftige Zahl zu reduzieren, werden die Inhalte von Geräteknoten nicht ausgegeben. Wenn Sie explizit die Inhalte mehrere Geräte ausgeben wollen, markieren Sie alle Geräteknoten, die an denen Sie interessiert sind, auf der rechten Seite der Konsole. Dannach starten Sie den Bericht oder Export.

8. Eigene Berichte

rxInventory enthält einen sehr leistungsfähigen Berichtsdesigner. Sie können neue Berichte vom `Berichte`-Knoten aus anlegen und diese in Ordnern verwalten. Sie finden mehrere Berichte im `Beispiel`-Ordner, die Sie als Vorlage nehmen können.

Innerhalb des Abfragedesigners können neue Eigenschaften durch das Anklicken des Eigenschaftsfeldes in der letzten Reihe angelegt werden.



Die Position von Eigenschaftszeilen kann durch Ziehen geändert werden. Um eine ausgewählte Spalte zu löschen, benutzen Sie bitte die **Entf**-Taste oder `Löschen` vom Kontext-Menü. Das `Alias`-Feld erlaubt es den Spaltennamen umzubenennen. Wenn `Alias` leer ist, wird der normale Name der Eigenschaft darstellt.

Wenn Sie die Reihenfolge der ausgegebenen Zeilen nach einem oder mehreren Eigenschaften sortieren wollen, können Sie die Richtung der Sortierung eines Properties mit `Sortierung` und die Reihenfolge der Sortierung durch `Sortierreihenfolge` auswählen.

Das `Gruppieren nach`-Feld erlaubt es sogenannte Aggregatfunktionen auf eine Eigenschaft anzuwenden. Die Ergebnismenge wird nach den übrigen Eigenschaften gruppiert und die ausgewählte Funktion (count, sum etc.) wird auf die ausgewählte Eigenschaft für jedes Gruppenelement angewendet.

Alle Abfragen und ihre Ordner können – einschließlich der Abfragen im `Beispiel`-Ordner – durch Kopieren und Verschieben neu angeordnet werden. Das kann entweder durch Ziehen mit der Maus oder durch Ausschneiden und Einfügen passieren. Beim Ziehen einer Abfrage oder Ordners mit der Maus ist die Standardaktion ihn an die neue Position zu verschieben. Wenn Sie ihn mit der rechten Maustaste ziehen, wird ein Menü dargestellt, aus dem Sie die gewünschte Funktion auswählen können. Wenn eine Abfrage oder Ordner auf eine andere Abfrage fallen gelassen wird, wird diese dort eingefügt und die andere Abfrage wird um eine Position nach unten verschoben. Wenn Sie eine Abfrage oder Ordner an das Ende einer Liste verschieben oder kopieren wollen, lassen Sie sie einfach auf den übergeordneten Ordner fallen.

8.1. Filter und Having

Im `Filter`-Reiter können Filterbedingungen auf die Ergebnismenge angewendet werden. Sie können komplexe Filterausdrücke erzeugen, indem Sie mehrere Predikate mit `UND`- und `ODER`-Operatoren verknüpfen. Mit Klammern können Ausdrücke gruppiert werden. Der Designer löscht bzw. fügt automatisch Klammer hinzu, um einen gleichwertigen Ausdruck zu erzeugen, der mit möglichst wenigen Klammern auskommt ohne sich dabei auf die Rangfolge der Operatoren verlassen zu müssen.

Der `Having`-Filter ist nur verfügbar wenn Aggregatfunktionen verwendet werden. Während der `Filter`-Ausdruck vor der Gruppierung angewendet werden, wirkt der `Having`-Ausdruck auf die finale Ergebnismenge. Dies ist auch der Grund, warum man im `Having`-Reiter nur Eigenschaften auswählen kann, die auch im Spaltenreiter ausgewählt sind und deshalb auch Teil der Ergebnismenge sind.

Anmerkung

Alle Zeichenkettenvergleiche sind unabhängig von Groß- und Kleinschreibung. Bei Datenbanken wie PostgreSQL und Oracle, die Groß- und Kleinschreibung unterscheiden, wird automatisch die Bedingung dafür intern umgeschrieben.

8.2. Verlauf

rxInventory führt Buch über Änderungen zwischen aufeinanderfolgenden Scans. Der Verlauf von Installation und Deinstallation von Software-Paketen auf einzelnen Rechnern kann innerhalb des Installationsverlauf-Knotens betrachtet werden. Für weitere Verlaufsinformationen können Sie eine Abfrage anlegen und innerhalb des Verlauf-Reiters den Zeitraum auswählen, der Sie interessiert.

Anmerkung

Bei der Abfrage von Verlaufsinformationen ist es wichtig, dass die erste Spalte von der richtigen Tabelle stammt. Wenn Sie zum Beispiel sehen wollen, wie sich der freie Plattenplatz entwickelt, muss als erste Spalte eine der `Partitions`-Eigenschaften ausgewählt werden und nicht z.B. der Gerätename. (Die `Gültig von`- und `Gültig bis`-Eigenschaften werden speziell behandelt und können unabhängig davon als erstes erscheinen.)

Hier gibt es auch einen Auswahlbox, mit der man archivierte Geräte in die Abfrage mit einschließen kann.

8.3. SQL Views

Um auf die Ergebnisse einer eigenen Abfrage von externen Applikationen aus zuzugreifen, können Sie eine darauf basierende Datenbank-View anlegen. Die View gibt die gleichen Datenfelder mit einigen Zusatzspalten aus. Die Namen dieser Extraspalten beginnen mit einem Unterstrich und können ignoriert werden.

Anmerkung

Microsoft SQL Server und die Microsoft Jet Engine unterstützen keine `ORDER BY`-Ausdrücke im definierenden Select-Ausdruck einer View. Deshalb müssen Sie die Ergebnisliste selbst sortieren lassen, wenn Sie Zeilen aus dem View abfragen.

Microsoft Sql Server Compact Edition unterstützt keine Views.

9. Berichtsbäume

Berichtsbäume werden wie normale Berichte angelegt. Anstatt aber nur eine einfache Liste anzuzeigen, wird die (fast) vollständige Baumstruktur aus dem rxInventory-Wurzelknoten unter ihm gespiegelt.

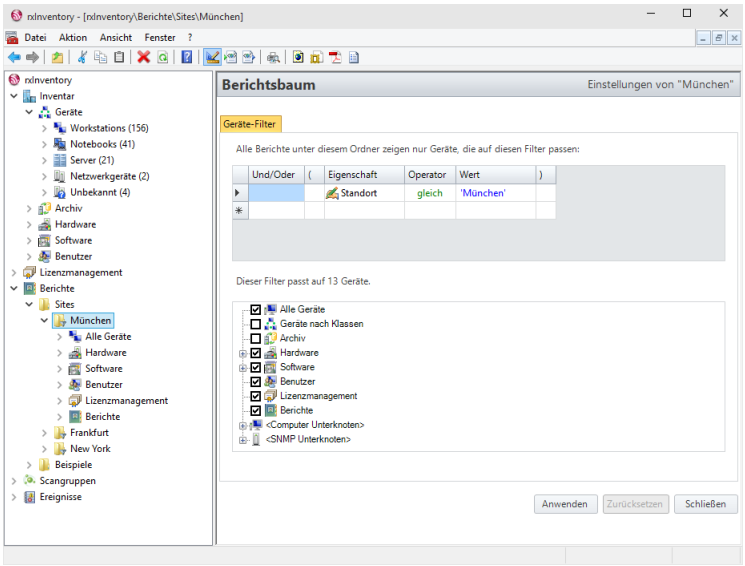
Es kann ein Filterausdruck angegeben werden, der die Menge der angezeigten Geräte einschränkt. Auf diese Weise ist es zum Beispiel möglich, nur mit den Computern und Geräten eines bestimmten Standorts, Organsiationseinheit, Subnetzes etc. zu arbeiten. Alle einwertigen Eigenschaften können für diesen Filter verwendet werden.

Der neu angelegte Berichtsbaum verhält sich wie eine Assetdatenbank, die nur die ausgefilterten Geräte enthält. Alle Statistiken und Summen werden entsprechend angepasst.

Innerhalb des Berichtsbaum-Designer ist es möglich bestimmte Knoten der Baumstruktur auszublenden oder einige anfänglich ausgeschlossene Knoten hinzuzufügen.

Jeder Berichtsbaum kann sein eigenes **Lizenz-Management** haben. Dieses Lizenzmanagement ist unabhängig vom Hauptlizenzmanagement und verwendet seine eigenen Softwarezähler.

Beim Anlegen eigener Berichte innerhalb eines Berichtsbaums wird die Filterbedingung des Berichtsbaums transparent auf den neu erzeugten Bericht vererbt. Sie können sogar einen Berichtsbaum innerhalb eines Berichtsbaums anlegen und so die Datenmenge weiter reduzieren.



10. Web Interface

rxInventory verfügt über einen integrierten Web-Server. Dieser ermöglicht es die inventarisierten Daten von einem anderen Computer abzurufen, ohne dass rxInventory auf diesem Rechner installiert werden muss.

Dieses Interface erlaubt nur lesenden Zugriff und kann passwortgeschützt werden. (Cookie müssen dafür eingeschaltet sein)

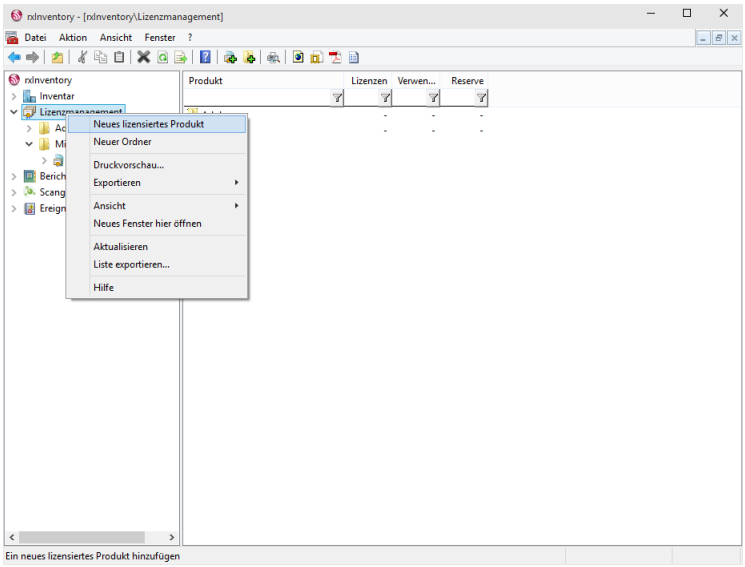
Der Web-Server läuft innerhalb des rxInventory-Dienstes. (Der gleiche Dienst, welcher für geplante Tasks ausführt.) Dieser Dienst muss laufen und mit einer Benutzerkennung eingerichtet sein, die Zugriff auf die Datenbank hat.

Das Web-Interface kann innerhalb der Hauptseite der rxInventory-Konsole konfiguriert werden.

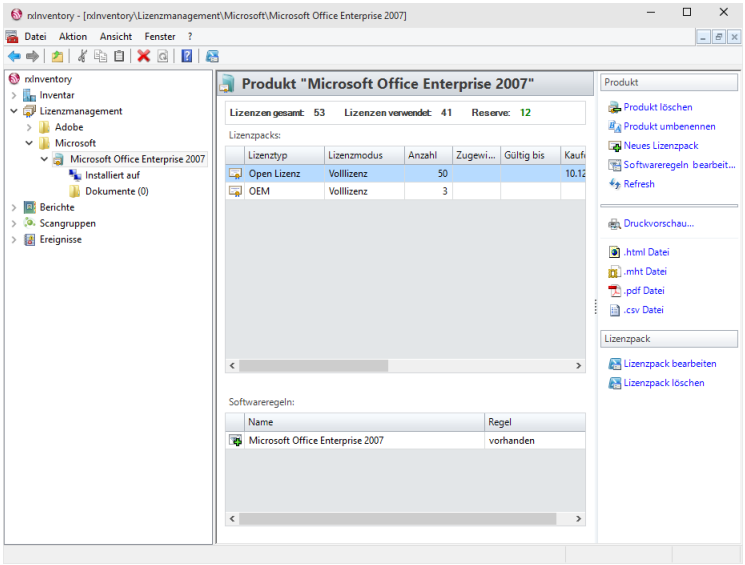
Kapitel 4. Software-Lizenzmangement

rxInventory enthält ein leistungsfähiges Lizenzmanagement. Es ermittelt automatisch die Anzahl der Softwarepakete, die auf allen Geräte installiert sind, und berechnet die benötigte Anzahl von Lizenzen. Indem diese Zahl mit der Anzahl der gekauften Lizenzen verglichen wird, können teuere Über- und gefährliche Unterlizenzierungen erkannt werden.

Ein neues lizensiertes Produkt kann durch einen Rechtsklick auf den `Lizenzmangement`-Knoten angelegt werden. Sie können dabei einen Namen angeben, der das Produkt identifiziert. (z.B. `Microsoft Office Enterprise 2007`)

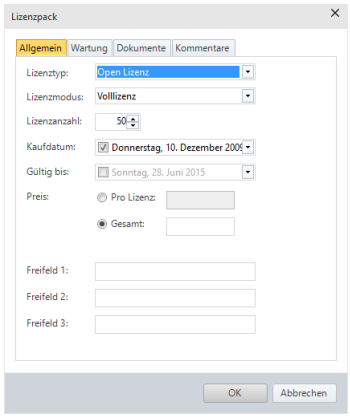


Hier sehen Sie ein Beispiel, wie die Produktseite für `Microsoft Office Enterprise 2007` aussehen könnte.



1. License Packs

Mindestens ein Lizenzpack sollte zu einem Produkt hinzugefügt werden. Das wichtigste Feld innerhalb des Lizenzpack-Formulars ist die Anzahl von Lizenzen. Die Summe der Lizenzen in allen nicht abgelaufenen Lizenzpacks wird ermittelt und im Header der Produktseite angezeigt. Ein Lizenzpack wird als abgelaufen betrachtet, wenn das `Gültig bis`-Feld einen Wert enthält, der älter als das aktuelle Datum ist.



2. Softwareregeln

Beim Berechnen der benötigten Lizenzen für ein Produkt, vergleicht rxInventory die erkannte Software auf jedem Computer mit den hier angegebenen Regeln. Es gibt vier Arten von Regeln:

Falls irgendeines dieser Softwarepakete auf einem Computer vorhanden ist, wird das Produkt als installiert betrachtet.

Nur wenn **alle** diese Softwarepakete auf einem Computer vorhanden sind, wird das Produkt als installiert betrachtet.

Falls irgendeines dieser Softwarepakete auf einem Computer vorhanden ist, wird das Produkt als **nicht** installiert betrachtet, selbst wenn andere Regeln zutreffen.

Wenn **alle** diese Softwarepakete auf einem Computer vorhanden sind, wird das Produkt als **nicht** installiert betrachtet, selbst wenn andere Regeln zutreffen.

3. Lizenzen Computern zuweisen

Innerhalb des `Installiert auf`-Knotens werden die Computer angezeigt, die eine Lizenz für das Produkt benötigen. (Über das Kontextmenü kann die Liste negiert werden, so dass alle Computer angezeigt werden, auf denen das Produkt nicht installiert ist.)

Produktlizenzen können einem Computer fest zugewiesen werden. Es ist auch möglich dabei ein Lizenzpaket zuzuordnen. Diese Computer werden dann mit einem grünen Haken in der Liste angezeigt. Falls ein Computer diesem Produkt zugewiesen wurde, aber eigentlich keine Lizenz mehr benötigt (z.B. wenn ein Paket gelöscht oder eine Softwareregel geändert wurde), erscheint es weiterhin in der Liste, aber das Computersymbol ist ausgegraut. Dieses Feature kann dazu benutzt werden unerwünschte Installationen und Deinstallationen eines Softwareprodukts zu erkennen.

Anmerkung

Der Zuweisungsstatus eines Computers hat nur Informationsfunktion und hat **keinen** Einfluss auf die Lizenzzählung. Das heißt auch, dass Sie einer Produktlizenz mehr Computer zuweisen können, als Lizenzen für das Produkt zur Verfügung stehen.

Kapitel 5. Eigene Skripts

1. Die Programmiersprache

Eigene Skripte ermöglichen dem Administrator zusätzliche Informationen zu den gescannten Geräten zu sammeln. Skripte können auch dazu genutzt werden, Eigenschaften des Standardscans zu ändern oder zu löschen. Eigene Skripte werden in einer Sprache geschrieben, die syntaktisch sehr ähnlich zu Visual Basic Script ist. rxInventory verwendet seinen eigenen Interpreter für diese Aufgabe, um Abhängigkeiten von der Scripting Runtime des Betriebssystems zu vermeiden und den Einfluss auf den Speicherbedarf und die Sicherheit zu minimieren. Es können keine externen Objekte (COM) erzeugt werden und nur eine begrenzte Anzahl von Funktionen stehen zur Verfügung. Für eine Referenz zur Visual Basic Skript-Sprache lesen Sie bitte in der Microsoft Dokumentation nach. Im folgenden finden Sie eine Liste der größten Unterschiede zwischen der rxInventory Skriptsprache und normalem VB-Skript:

- Alle Variablen müssen vor der ersten Benutzung definiert werden. Es gibt keinen `OPTION EXPLICIT`-Befehl.
- Es gibt nur die folgenden Datentypen: String, Long, Float, Date, Arrays and Object. Zusätzlich kann eine Variable leer (empty) sein.
- Variablen vom long-Typ werden intern durch eine vorzeichenbehaftete 64-bit Integer repräsentiert.
- Nur eine begrenzte Untermenge der VB-Standardfunktionen steht zur Verfügung.

2. Struktur der Skripts

Der Körper (main body) eines Skripts sollte nur für grundlegende Initialisierungsvorgänge genutzt werden. Weil dieser sehr früh im Scanprozess verarbeitet wird, besteht zu diesem Zeitpunkt noch keine Verbindung zum Zielgerät.

Eigener Code sollte in eine der folgenden Unterroutinen gesetzt werden, welche automatisch während des Scanprozesses für jedes Script aufgerufen werden. Für jede `BeforeXXX`-Funktion gibt es eine `AfterXXX`-Funktion, die sich nur darin unterschiedet, dass Sie nach dem Scan der Standardeigenschaften ausgeführt wird.

`BeforeScan()` — Wird aufgerufen, nachdem der Zielrechner angebunden ist und alle Initialisierungen stattgefunden haben, aber bevor die Standardeigenschaften erfasst werden. Dies ist die empfohlene Funktion, um eigenen Code beim Scannen von Windowsrechnern auszuführen.

`BeforeUserScan()` — Diese Funktion wird für jeden Benutzer aufgerufen, der gerade auf dem Zielrechner angemeldet ist. Eigenschaften, die in dieser Funktion gesetzt werden, erscheinen innerhalb des Benutzer-Unterknotens des Computers. Zugriffe auf den `HKEY_CURRENT_USER` Registry-Hive werden automatisch auf den gerade gescannten Benutzer umgelenkt. Dies ist die empfohlene Funktion, um eigenen Code beim Scannen von Benutzereigenschaften bei Windowsrechnern auszuführen.

Anmerkung

Der Aufruf von SNMP- und SSH-Funktionen innerhalb von `BeforeScan()` und `BeforeUserScan()` wird nicht unterstützt.

`BeforeSnmpScan()` Diese Funktion wird ausgeführt, wenn das Ziel als SNMP-Gerät identifiziert wurde. Die Ausführung erfolgt bevor die Standard-SNMP-Eigenschaften erfasst werden.

Anmerkung

Der Aufruf von Registry-, Active Directory- und WMI-Funktionen innerhalb von `BeforeSnmpScan()` wird nicht unterstützt.

`BeforeLinuxScan()` Wird aufgerufen, wenn das Ziel über SSH gescannt wird und als Linux erkannt wurde.

`BeforeMacScan()` Wird aufgerufen, wenn das Ziel über SSH gescannt wird und als Mac erkannt wurde.

Anmerkung

Der Aufruf von Registry-, WMI-, ActiveDirectory- und SNMP-Funktionen innerhalb von `BeforeLinuxScan()` and `BeforeMacScan()` wird nicht unterstützt.

`On`*`ItemType`*`(`*`properties`*`)` Diese Art von Funktion wird für jeden Eintrag vom Typ *`ItemGroup`* aufgerufen, wenn dieser ausgegeben wird. Wenn `false` zurückgegeben wird, wird der Eintrag verworfen. Der Parameter *`properties`* ist ein Objekt, das die Eigenschaften des Eintrags enthält. Eine Liste alle Eintragstypen und Eigenschaften finden Sie unter Abschnitt 5, „Callback-Funktionen“.

Anmerkung

Zurzeit kann diese Art von Code nicht mit dem `Test`-Button überprüft werden. Sie müssen die Funktionsfähigkeit Ihres Codes durch einen normalen Scan überprüfen. Wir empfehlen eine neue Scangruppe mit nur einem einzelnen Computer als Testumgebung anzulegen.

Beispiel:

```
Function OnSoftware(software)
  If software.Name = "Microsoft SQL Server 2008" Then
    Dim key
    For Each key In RegEnumKeys("HKLM\SOFTWARE\Microsoft\Microsoft SQL Server")
      If key.Name like "MSSQL10.*" Then
        Dim productkey
        productkey=RegGetValue(key.Path & "\Setup", "DigitalProductID", rxDigitalProductID)
        If Len(productkey)>0 Then
          software.LicenseKey=productkey
          Exit For
        End If
      End If
    Next
  End If
```

```
End Function
```

3. VB-Funktionsreferenz

3.1. Ausgabefunktionen

AddSoftware

`AddSoftware` *`name[, key[, version[, publisher[, installlocation[, licensekey[, licenseid[, uninstall[, installdate[, installsource[, frequency[, lastused[, size]]]]]]]]]]]`*

Fügt ein Softwarepaket in die Inventardaten ein.

Beispiel:

```
AddSoftware "My Special Software", "", "3.1"
```

SetProperty

`SetProperty` *`name, value`*

Schreibt eine eigene String-Eigenschaft in die Ausgabe. In der Datenbank wird beim Einfügen eine String-Spalte angelegt

Beispiel:

```
SetProperty "IE Version", GetFileVersion("shdocvw.dll")"
```

SetNumberProperty

`SetNumberProperty` *`name, value`*

Schreibt eine eigene Eigenschaft in die Ausgabe und kennzeichnet sie als Zahl. In der Datenbank wird beim Einfügen eine Zahlenspalte angelegt.

SetIntegerProperty

`SetIntegerProperty` *`name, value`*

Schreibt eine eigene Eigenschaft in die Ausgabe und kennzeichnet sie als Integer. In der Datenbank wird beim Einfügen eine Integerspalte angelegt.

SetDateProperty

`SetDateProperty` *`name, value`*

Schreibt eine eigene Eigenschaft in die Ausgabe und kennzeichnet sie als Datum. In der Datenbank wird beim Einfügen eine Datumsspalte angelegt.

AbortScan

`AbortScan [`*`reason`*`]`

Bricht den Scan ab. Dies ist nützlich, falls ein Gerät ermittelt wurde, welches nicht gescannt werden soll. Es kann auch dazu verwendet werden, einen unerwarteten Zustand anzuzeigen. Der *`reason`* wird zu den Fehlerinformationen der Ausgabe geschrieben.

3.2. Windows-Informationsfunktionen

GetEnvironmentVariable

`GetEnvironmentVariable(`*`variablename`*`)`

Liefert eine Variable aus der Umgebung des gescannten Rechners zurück.

GetComputerName

`GetComputerName()`

Liefert den Namen des gescannten Computers oder Gerätes zurück.

IsX64

`IsX64`

Liefert `true` zurück, falls eine 64-bit Plattform gescannt wird.

IsOs9x

`IsOs9x`

Liefert `true` zurück, falls ein Gerät mit einem Windows 95/9x/Me-Betriebssystem gescannt wird.

IsOsUnix

`IsOsUnix`

Liefert `true` zurück, falls ein Gerät ohne einem Microsoft Windows-Betriebssystem gescannt wird, das aber SMB unterstützt.

GetOsMajorVersion

`GetOsMajorVersion`

Ermittelt die Major-Version des Betriebssystems.

GetOsMinorVersion

`GetOsMinorVersion`

Ermittelt die Minor-Version des Betriebssystems.

GetServicePackDisplayName

`GetServicePackDisplayName`

Ermittelt das Servicepack des Betriebssystems.

GetScannedIp

`GetScannedIp`

Liefert die IP-Adresse zurück, mit der der Scanner verbunden ist. Die IP `127.0.0.1` wird zurückgegeben, wenn ein lokaler Rechner mit Hilfe von `rxscan.exe` gescannt wird. (z.B. in einem Logonskript)

3.3. WMI-Funktionen

WmiQuery

`WmiQuery(`*`wql`*`)`

Führt die Abfrage `wql` aus und liefert ein Array von WMI-Objekten zurück. WMI-Objekte sind im Grunde Dictionaries mit Schlüssel-Wert-Paaren für jedes Property. Falls die Abfrage fehlschlägt, wird ein leeres Array zurückgeliefert. Dies ist auch der Fall, wenn WMI nicht zur Verfügung steht oder der Parameter `NoWmi` angegeben wurde.

Beispiel:

```
Function BeforeScan()
  Dim str
  Dim w
  For Each w In WmiQuery("SELECT * FROM Win32_PageFile")
    If Len(str)>0 Then
      str=str & ","
    End If
    str=str & w("EightDotThreeFileName") & ":" & CLng(w("FileSize")/1024/1024) & "MB"
  Next
  SetProperty "Page Files", str
End Function
```

WmiSingleQuery

`WmiSingleQuery` *`wql`*

Führt die Abfrage `wql` aus und liefert das erste WMI-Objekt zurück. Falls die Abfrage kein Objekt liefert oder die Abfrage fehlschlägt, wird ein leeres WMI-Objekt zurückgeliefert.

Beispiel:

```
Dim w
Set w=WmiQuerySingle("SELECT * FROM Win32_ComputerSystem")
SetProperty "Is part of a domain", w("PartOfDomain")
```

3.4. Registryfunktionen

Für die meisten Registryfunktionen muss der Name eines Registry-Schlüssels angegeben werden. Das Format ist das gleiche wie beim Kopieren einen Schlüssel aus `regedit.exe`. Ein Beispiel ist `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion`. Unterstützte Root-Schlüssel sind:

- HKEY_LOCAL_MACHINE (oder HKLM)
- HKEY_CURRENT_USER (oder HKCU)
- HKEY_USERS (oder HKU)

Wenn 32 an den Schlüssel angehängt wird (zum Beispiel HKLM32\Software), wird der 32-bit WoW-Schlüssel eines 64-bit Betriebssystems abgefragt. Auf 32-bit Betriebssystemen wird der Standardschlüssel verwendet.

Der Schlüssel auf den bei der Angabe von HKEY_CURRENT_USER zugegriffen wird, hängt vom Zustand ab. Die BeforeUserScan()- und AfterUserScan()-Funktionen werden für jeden Benutzer aufgerufen, der aktuell auf dem System angemeldet ist. Wird HKEY_CURRENT_USER außerhalb einer dieser zwei Funktionen benutzt, wird auf den Hive des Benutzers zugegegriffen, der gerade auf der Computer-Konsole angemeldet ist.

RegGetValue

RegGetValue(*registrypath, valuename*[, *flags*])

Liefert einen Wert aus der Registry zurück. Verwenden Sie einen leeren *valuename*, um auf den Standardwert des Schlüssels zuzugreifen. Falls der Wert oder Schlüssel nicht existiert, wird ein leerer (empty) Wert zurückgeliefert. Der *flags*-Parameter ist optional.

Die folgenden Konstanten sind definiert und können mit dem &-Operator kombiniert werden:

rxUnixTime	Liefert einen Datums/Zeit-Wert zurück, in dem ein Zahlenwert als die Zahl der Sekunden interpretiert wird, die seit Mitternacht UTC des 1. Januar 1970 vergangen sind.
rxDateTime	Wandelt eine Zeichenkette in einen Datums/Zeit-Wert um.
rxBinary	Ein Array von Byte-Werten wird zurückgeliefert. Falls der Registrywert vom Typ REG_SZ, REG_EXPAND_SZ or REG_MULTI_SZ ist, werden Unicode-Bytes zurückgeliefert, inklusive der Null am Ende.
rxDigitalProductID	Eine Microsoft DigitalProductID wird in das XXXXX-XXXXX-XXXXX-XXXXX-XXXXX Format konvertiert.
rxInteger	Falls der Registry-Wert ein String ist, wird er in eine Zahl umgewandelt.
rxString	Ein Registrywert vom Typ REG_BINARY wird als Zeichenkette interpretiert.
rxMultiSzAsArray	Standardmäßig wird jeder Wert eines REG_MULTI_SZ-Werts an eine mit Pipe-Symbolen getrennte Zeichenkette angefügt. Mit diesem Flag wird dagegen ein Array von Werten zurückgeliefert.

Beispiel:

```
SetProperty "Windows Product ID", RegGetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVers
    "ProductId")
```

RegOpenKey

RegOpenKey(*registrypath*)

Öffnet *registrypath* und liefert ein Registryschlüssel-Objekt zurück. Falls der Schlüssel nicht existiert, wird Nothing zurückgegeben.

Ein Registryschlüssel-Objekt stellt folgende Eigenschaften und Methoden zur Verfügung:

`key.Name`	Der Name des Schüssels
`key.Path`	Der vollständige Pfad des Schlüssels
`key.GetValue(`*`valuename`*`)`	Siehe `RegGetValue()`.
`key.Open(`*`keyname`*`)`	Öffnet den Unterschlüssel *`keyname`* des Schlüssels. `Nothing` wird zurückgeliefert, falls der Name nicht existiert.
`key.Keys`	Liefert ein Array mit allen Unterschlüssel-Objekten zurück.
`key.KeyNames`	Liefert ein Array von allen Unterschlüssel-Namen zurück.
`key.ValueNames`	Liefert ein Array mit den Wertnamen innerhalb des Schlüssels zurück.

RegEnumKeys

`RegEnumKeys(`*`registrypath`*`)`

Zählt alle Unterschlüssel auf und gibt ein Array von Schlüsselobjekten zurück. Das Array ist leer, falls der Registrypfad nicht existiert oder nicht geöffnet werden kann. Siehe `RegOpenKey()` für eine Beschreibung der Schlüsselobjekte.

Beispiel:

```
Dim key
Dim keylist
keylist=""
For Each key In RegEnumKeys("HKLM\Software")
    If Len(keylist)>0 Then
        keylist=keylist & ","
    End If
    keylist=keylist & key.Name
Next
SetProperty "Software Sub-Keys", keylist
```

RegEnumKeyNames

`RegEnumKeyNames(`*`registrypath`*`)`

Liefert ein Array der Unterschlüsselnamen innerhalb von *`registrypath`* zurück.

RegEnumValueNames

`RegEnumValueNames(`*`registrypath`*`)`

Liefert ein Array der Wertnamen innerhalb *`registrypath`* zurück.

3.5. Dateifunktionen

Viele Dateifunktionen benötigen einen Pfad als Parameter. Pfadnamen sind relativ zum SYSTEM32-Verzeichnis. Verwenden Sie `..` um auf das Windows-Verzeichnis zuzugreifen. Absolute Dateinamen müssen mit Laufwerksbuchstaben angegeben werden, wie Sie auf der Zielmaschine verwendet würden. UNC-Pfadnamen sind nicht erlaubt. Dateifunktionen funktionieren nur mit Rechnern, die die administrativen Freigaben eingeschaltet haben. (z.B. C$,D$ ...)

GetFileVersion

`GetFileVersion(`*`path`*`)`

Liefert die Versionsnummer aus den Ressourcen der ausführbaren Datei oder dynamischen Programmbibliothek zurück. Falls ein Fehler auftritt, wird ein leerer (empty) Wert zurückgeliefert.

GetFileLanguage

`GetFileLanguage(`*`path`*`)`

Liefert die Sprachressource aus der Datei zurück.

GetFileAttributes

`GetFileAttributes(`*`path`*`)`

Liefert die Attribute der Datei zurück. -1 wird zurückgeliefert, falls die Datei nicht existiert oder ein Fehler aufgetreten ist.

Die folgenden Konstanten stehen zur Verfügung:

`rxReadOnly`	1
`rxHidden`	2
`rxSystem`	4
`rxDirectory`	16
`rxArchive`	32
`rxDevice`	64
`rxNormal`	128
`rxTemporary`	256
`rxSparseFile`	512
`rxReparsePoint`	1024
`rxCompressed`	2048
`rxOffline`	4096
`rxNotContentIndexed`	8192
`rxEncypted`	16384

GetFileSize

`GetFileSize(`*`path`*`)`

Liefert die Größe der Datei zurück. -1 wird zurückgeliefert, falls die Datei nicht existiert oder ein Fehler aufgetreten ist.

GetFileTime

`GetFileTime(`*`path`*`[, `*`whichtime`*`])`

Liefert den Zeitpunkt zurück, wann eine Datei erzeugt, geschrieben oder auf sie zuletzt zugegriffen wurde. Ein leerer (empty) Wert wird zurückgeliefert, falls die Datei nicht existiert. Falls der *whichtime*-Parameter nicht angegeben wird, wird `rxFileTimeWrite` angenommen.

Die folgenden Konstanten stehen für den Parameter *whichtime* zur Verfügung:

`rxFileTimeWrite`	Liefert den Zeitpunkt der letzten Änderung der Datei zurück.
`rxFileTimeCreation`	Liefert den Zeitpunkt zurück, zu dem die Datei angelegt wurde.
`rxFileTimeAccess`	Liefert den Zeitpunkt des letzten Zugriffs auf die Datei zurück.

FileExists

`FileExists(`*path*`)`

Liefert `true` zurück, wenn die Datei existiert.

DirectoryExists

`DirectoryExists(`*path*`)`

Liefert `true` zurück, wenn das Verzeichnis existiert.

3.6. Active Directory-Funktionen

GetADComputerObject

`GetADComputerObject`

Verbindet sich mit einem Domain-Controller der Domäne des gescannten Rechners und ermittelt die Attribute des zu dem gescannten Rechner gehörenden Computer-Objekts. Bei Erfolg wird ein Computer-Object zurückgegeben. `Nothing` wird zurückgeliefert, falls der Computer nicht Teil einer Active Directory-Domäne ist.

Das zurückgegebene Object stellt folgende Funktion zur Verfügung:

`co.Get(`*name*`)` Liefert den Wert des benannten Attributs.

Auf Attribute kann auch über gleichnamige Properties des Objekts zugegriffen werden.

Beispiel:

```
Dim computer
Set computer=GetADComputerObject
SetProperty "Computer Description", computer.Description
```

GetADUserObject

`GetADUserObject`

Verbindet sich mit einem Domain-Controller der Domäne des gescannten Rechners und ermittelt die Attribute des zu dem angemeldeten Benutzer gehörenden Benutzer-Objekts. Bei Erfolg wird ein Benutzer-Object zurückgegeben. `Nothing` wird zurückgeliefert, falls der Computer nicht Teil einer Active Directory-Domäne ist. Die Funktion darf nur innerhalb einer Benutzerscanfunktion (z.B.

`BeforeUserScan()`) aufgerufen werden. Die erfassten Eigenschaften erscheinen im Gerät deshalb auch unterhalb des jeweiligen Benutzerknotens.

Das zurückgegebene Object stellt folgende Funktion zur Verfügung:

`co.Get(`*`name`*`)` Liefert den Wert des benannten Attributs.

Auf Attribute kann auch über gleichnamige Properties des Objekts zugegriffen werden.

Beispiel:

```
Function BeforeUserScan()
  Dim u
  Set u=GetADUserObject
  If Not u Is Nothing Then
    SetProperty "Telefon", u.telephoneNumber
    SetProperty "EMail", u.mail
  End If
End Function
```

3.7. SNMP-Funktionen

SnmpGetValue

`SnmpGetValue(`*`oidstring`*`[, `*`hint`*`])`

Ermittelt einen SNMP-Wert über den angegebenen *`oidstring`*. Im Fehlerfall, zum Beispiel wenn die OID nicht existiert, wird ein leerer (empty) Wert zurückgegeben.

Der Parameter *`hint`* ist optional und kann folgende Konstanten haben:

`rxSnmpHex`	Liefert eine Zeichenkette mit hexadezimalen Bytes, die den Wert repräsentieren, zurück.
`rxSnmpMac`	Interpretiert den Wert als MAC-Adresse.
`rxSnmpDateAndTime`	Interpretiert den empfangen Wert als Snmp DateAndTime-Wert.

Beispiel:

```
SetProperty "SNMP Location", SnmpGetValue(".1.3.6.1.2.1.1.6.0")
```

SnmpGetNextValue

`SnmpGetNextValue(`*`oidstring, nextoidstring`*`)`

Wie `SnmpGetValue`. Die nächste OID wird in den *`nextoidstring`*-Parameter geschrieben.

3.8. SSH-Funktionen

SshExecute

`SshExecute(`*`command`*`)`

Führt *`command`* über eine SSH-Verbindung aus und gibt ein result-Objekt zurück.

Ein SSH-Result-Objekt stellt folgende Eigenschaften zur Verfügung:

`result.ExitCode` Der Fehlerwert des Befehls

`result.StdOut` Ein Stream, der die Standardausgabe des Befehls enthält.

`result.StdErr` Ein Stream, der die Standardfehlerausgabe des Befehls enthält.

Ein Stream von *`StdOut`* oder *`StdErr`* stellt folgende Eigenschaften oder Methoden zur Verfügung.

`stream.EndOfFile` `true`, falls das Ende des Streams erreicht wurde.

`stream.ReadLine` Gibt die nächste Zeile aus dem Stream zurück.

`stream.SeekToBeginning` Stellt den Stream auf den Anfang zurück.

Beispiel:

```
Dim result
Set result=SshExecute("cat /etc/passwd|wc -l")
If result.ExitCode=0 Then
  If Not result.StdOut.EndOfFile Then
    SetProperty "Lines in passwd file", result.StdOut.ReadLine
  End If
End If
```

3.9. VB-Funktionen

Asc

`Asc(`*`value`*`)`

Liefert den Zeichencode für das erste Zeichen von *`value`* zurück.

CDate

`CDate(`*`variablename`*`)`

Wandelt einen String in ein Datum um. Falls das Datum nicht gelesen werden kann, wird das Datum 2000-01-01 zurückgegeben.

CDbl

`CDbl(`*`value`*`)`

Wandelt den angegebenen Wert in eine Fließkommazahl um.

Chr

`Chr(`*`code`*`)`

Liefert das Zeichen mit dem angegebenen Code zurück.

CLng

`CLng(`*`value`*`)`

Wandelt den angegebenen Wert in eine Zahl um.

CStr

`CStr(value)`

Wandelt den angegebenen Wert in eine Zeichenkette um.

Day

`Day(date)`

Liefert den Tag von *date* zurück.

DateSerial

`DateSerial(year, month, day)`

Gibt das angegebene Datum zurück.

Env

`Env(variablename)`

Liefert eine Variable aus der lokalen Umgebung zurück. Verwenden Sie die `GetEnvironmentVariable` -Funktion um eine Umgebungsvariable vom gescannten Rechner zu erhalten.

Hour

`Hour(date)`

Liefert die Stunde von *date* zurück.

IIf

`IIf(condition, truevalue, falsevalue)`

Liefert *truevalue* zurück, falls *condition* wahr ist. Andernfalls wird *falsevalue* zurückgegeben.

InStr

`InStr([start, ]string, find)`

Sucht nach dem ersten Vorkommen von *find* in *string* beginnend mit *start*. Wenn *start* weggelassen wird, dann wird am Anfang der Zeichenkette begonnen.

InStrRev

`InStrRev([start, ]string, find)`

Sucht nach dem letzten Vorkommen von *find* in *string* beginnend mit *start*. Wenn *start* weggelassen wird, dann wird beim Ende der Zeichenkette begonnen.

IsArray

`IsArray(value)`

Gibt `true` zurück, falls *`value`* ein Array ist.

IsEmpty

`IsEmpty(`*`value`*`)`

Gibt `true` zurück, falls *`value`* ein leerer (empty) Wert ist. Ein leerer (empty) Wert ist nicht das gleiche wie eine leere Zeichenkette. Viele Funktionen, wie zum Beispiel `RegGetValue` liefern einen leeren Wert zurück, falls ein Fehler auftritt.

IsNumeric

`IsNumeric(`*`value`*`)`

Überprüft, ob *`value`* in eine Zahl umgewandelt werden kann.

Join

`Join(`*`array, delimiter`*`)`

Verbindet die Zeichenketten des *`array`* getrennt durch *`delimiter`*.

LBound

`LBound(`*`array`*`)`

Gibt die untere Grenze eines Arrays zurück. Normalerweise Null.

Left

`Left(`*`string, length`*`)`

Liefert die ersten *`length`*-Zeichen einer Zeichenkette zurück.

Len

`Len(`*`string`*`)`

Gibt die Länge des *`string`* zurück.

Mid

`Mid(`*`string, start`*`[, `*`length`*`])`

Liefert die *`length`* Zeichen des *`string`*, beginnend bei der Position *`start`*. Falls *`length`* weggelassen wird, werden alle Zeichen bis zum Ende zurückgegeben.

Minute

`Minute(`*`date`*`)`

Liefert die Minuten von *`date`* zurück.

Month

`Month(`*`date`*`)`

Liefert den Monat von *date* zurück.

MsgBox

```
MsgBox text
```

Zeigt eine Dialogbox mit *text* an. Diese Unterfunktion ist für Testzwecke bei der Entwicklung von Skripts gedacht. Sie sollte nicht in Produktionsumgebungen benutzt werden.

Now

```
Now
```

Liefert das aktuelle Datum und Zeit.

Replace

```
Replace(string, find, replace)
```

Ersetzt jedes Vorkommen von *replace* in *string* und liefert das Ergebnis zurück.

Right

```
Right(string, length)
```

Liefert die *length* letzten Zeichen von *string* zurück.

Second

```
Second(date)
```

Liefert die Sekunden von *date* zurück.

Split

```
Split(string, delimiter[, count])
```

Zerteilt *string* an *delimiter* und liefert ein Array von Strings zurück. *count* begrenzt die maximale Anzahl der Array-Elemente.

TimeSerial

```
TimeSerial(hour, minutes, seconds)
```

Gibt die angegebene Zeit zurück.

Trim

```
Trim(string)
```

Schneidet Leerräume links und rechts von *string* ab.

UBound

```
UBound(array)
```

Liefert die Obergrenze eines Arrays zurück. Dies ist normalerweise die Zahl der Elemente im Array minus eins.

Year

`Year(`*`date`*`)`

Liefert das Jahr von *`date`* zurück.

Weekday

`Weekday(`*`date`*`)`

Liefert den Wochentag von *`date`* zurück. Beginnend mit Sonntag = 1.

4. Spezielle Funktionen

4.1. NewDictionary

`NewDictionary()`

Legt ein neues Dictionary-Objekt an. Das zurückgegebene Objekt ist sehr ähnlich zu der vom Microsoft Script Host bekannten Klasse `Scripting.Dictionary`. Die Schlüssel im Dictionary-Objekt werden als Zeichenketten behandelt. Bei Vergleichen wird die Groß- und Kleinschreibung nicht berücksichtigt.

Ein Dictionary-Objekt stellt folgende Methoden und Felder zur Verfügung:

`dict.Keys`	Liefert ein Array aller Schlüssel-Elemente.
`dict.Items`	Liefert ein Array aller Einträge zurück.
`dict.Item(`*`key`*`)`	Liefert den Eintrag, der mit dem Schlüssel *`key`* assoziiert ist, zurück.
`dict.Add` *`key, value`*	Fügt den Eintrag dem Dictionary hinzu.
`dict.Exists(`*`key`*`)`	Gibt `true` zurück, wenn der angegebene Schlüssel existiert.
`dict.Remove` *`key`*	Löscht das Element aus dem Dictionary.
`dict.RemoveAll`	Löscht alle Einträge aus dem Verzeichnis.
`dict.Count`	Gibt die Anzahl der Einträge im Dictionary zurück.

5. Callback-Funktionen

5.1. Eintrags-Callbacks

Bevor ein inventarisierter Eintrag der Datenausgabe hinzugefügt wird, wird eine Funktion innerhalb des Skripts überprüft. Falls die Funktion existiert und einen `false` Wert zurückgibt, wird der Eintrag nicht zu

den Inventarisierungsdaten hinzugefügt. Zusätzlich können einzelne Eigenschaften des Eintrags überprüft und geändert werden.

OnUser

`OnUser(`*`user`*`)`

Objekt *`user`* hat die folgenden Eigenschaften:

`user.ID`	User ID (Short String (64 characters))
`user.Name`	User Name (String)
`user.Domain`	User Domain (String)
`user.FullName`	User Full Name (String)
`user.Session`	User Session (String)
`user.LogonServer`	User Logonserver (String)
`user.LastLogonTime`	User Last Logon Time (Date/Time)

OnNetworkAdapter

`OnNetworkAdapter(`*`adapter`*`)`

Objekt *`adapter`* hat die folgenden Eigenschaften:

`network.Name`	Network Adapter (String)
`network.IpAddress`	IP Address (IP Address)
`network.MacAddress`	MAC Address (Short String (64 characters))
`network.SubnetMask`	Subnet Mask (IP Address)
`network.Network`	Network (IP Address)
`network.DefaultGateway`	Default Gateway (IP Address)
`network.DnsServer`	DNS Server (IP Address)
`network.DnsDomain`	DNS Domain (String)
`network.WinsServer`	WINS Server (IP Address)
`network.DhcpServer`	DHCP Server (IP Address)
`network.Speed`	Speed (Integer)

OnSoftware

`OnSoftware(`*`software`*`)`

Objekt *`software`* hat die folgenden Eigenschaften:

`software.Name`	Software Name (String)
`software.Key`	Software Key (String)

`software.Version`	Software Version (String)
`software.Publisher`	Software Publisher (String)
`software.InstallLocation`	Software Install Location (String)
`software.LicenseKey`	Software License Key (String)
`software.LicenseId`	Software License ID (String)
`software.Uninstall`	Software Uninstall (String)
`software.InstallDate`	Software Install Date (Date/Time)
`software.InstallSource`	Software Install Source (String)
`software.Frequency`	Software Frequency (Integer)
`software.LastUsed`	Software Last Used (Date/Time)
`software.Size`	Software Size (Integer)

OnHotfix

`OnHotfix(`*`hotfix`*`)`

Objekt *`hotfix`* hat die folgenden Eigenschaften:

`hotfix.Name`	Hotfix Name (String)
`hotfix.Key`	Hotfix Key (String)
`hotfix.Product`	Hotfix Product (String)
`hotfix.Type`	Hotfix Type (String)
`hotfix.InstallDate`	Hotfix Install Date (Date/Time)
`hotfix.InstalledBy`	Hotfix Installed By (String)
`hotfix.Uninstall`	Hotfix Uninstall String (String)

Kapitel 6. Automatisierung von Vorgängen

1. InvCmd.exe Kommandozeilenparameter

InvCmd.exe Kommandozeilenoptionen:

InvCmd.exe –dropall

Löscht alle Geräte aus der Datenbank. Eigene Abfragen und die Definition (nicht die Daten) von benutzerdefinierten Eigenschaften werden behalten.

–delete *devicename* [–force]

Löscht das Geräte mit dem Namen *devicename* aus der Datenbank. Falls mehrere Geräte mit dem gleichen Namen gefunden werden, muss der Schalter *–force* angegeben werden, um sie alle zu löschen.

–setuserdefinedproperty *devicename propertyname value* [–force]

Setzt die benutzerdefinierte Eigenschaft *propertyname* des Geräts *devicename* auf *value*. Falls mehrere Geräte mit dem gleichen Namen gefunden werden, muss der Schalter *–force* angegeben werden, um das Property auf sie alle zu setzen.

–dropuserdefinedproperty *propertyname*

Löscht die Spalte der benutzerdefinierten Eigenschaft *propertyname* aus der Datenbank. Falls Sie die Eigenschaft nur von einem einzelnen Gerät löschen wollen, benutzen Sie bitte *–setuserdefinedproperty* mit einer leeren Zeichenkette als *value*.

–backup *dumpfile*

Speichert die Geräte aus der Datenbank in eine Datei. Sie sollten die Erweiterung `.rxd` für die Datei verwenden.

–restore *dumpfile* [–mode create]

Liest die Geräte aus der Datenbank wieder in die Datenbank zurück.

Warnung

Standardmäßig werden alle Geräte aus der Datenbank gelöscht, bevor die Dumpdatei importiert wird. Falls Sie die Geräte zusammenführen wollen, müssen Sie den *–mode create* Parameter angeben. Die kann zu doppelten Namen führen, falls Dateien mit dem gleichen Namen in der Dumpdatei und der Datenbank existieren.

–scan

Startet alle Scangruppen. Dies ist gedacht, um Scans mit dem Windows Task Scheduler zu automatisieren. Bitte stellen Sie sicher, dass der Prozess unter dem richtigen administrativen Account läuft, um Netzwerkzugriff auf die gescannten Geräte zu ermöglichen.